5. Boolean indexing

Boolean indexing in Pandas refers to the process of filtering data in a DataFrame or Series using Boolean conditions.

When apply a Boolean condition to a DataFrame or a Series, it returns a Series of Boolean values that correspond to whether each element meets the condition.
Ex:

```
df = pd.DataFrame({
        'A': [10, 20, 30, 40, 50],
        'B': [5, 15, 25, 35, 45]
})
```

```
8    print(df)

PROBLEMS    TERMINAL    ···                powershell
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
    A   B
0  10   5
1  20  15
2  30  25
3  40  35
4  50  45
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
```

1. Let's Boolean index the `A` column for values less than or equal to 25

```
8    condition = df['A'] <= 25
9    print(condition)
10   print(type(condition))

PROBLEMS    TERMINAL    ···                powershell
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
0     True
1     True
2    False
3    False
4    False
Name: A, dtype: bool
<class 'pandas.core.series.Series'>
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
```

- Since we only index the `A` column, `condition` is in type of "Series" and it consist of Boolean values.

2. Let's Boolean index the whole DataFrame for values less than or equal to 25.

```
condition = df[['A', 'B']] <= 25
print(condition)
print(type(condition))
```

```
PROBLEMS    TERMINAL    ...                powershell

PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>

       A      B
0   True   True
1   True   True
2  False   True
3  False  False
4  False  False
<class 'pandas.core.frame.DataFrame'>
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
```

- Now it returns a DataFrame full of Boolean values indicating if each data element has satisfied the given Boolean expression or not.

The returning Series/DataFrame after Boolean indexing is called the **"Boolean mask"**.

Once we done with the Boolean indexing, we can use that Boolean Mask to filter out the values from the original DataFrame.

```
filtered_df = df[condition]
```

```
condition = df['A'] <= 25
print(condition)
filtered_df = df[condition]
print(filtered_df)
```

```
PROBLEMS    TERMINAL    ...                powershell

PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
0     True
1     True
2    False
3    False
4    False
Name: A, dtype: bool
    A   B
0  10   5
1  20  15
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
```

- It only returns the rows with only `A` has values less or equal to 25.

Let's take only rows with `A` has values greater than 25.

```
condition = df['A'] > 25
print(condition)
filtered_df = df[condition]
print(filtered_df)
```

```
PROBLEMS    TERMINAL    ...                powershell

PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
0    False
1    False
2     True
3     True
4     True
Name: A, dtype: bool
    A   B
2  30  25
3  40  35
4  50  45
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
```

Let's take the only row with column `B` has value 35.

```
condition = df['B'] == 35
print(condition)
filtered_df = df[condition]
print(filtered_df)
```

```
PROBLEMS    TERMINAL    ...                powershell

PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals> py
0    False
1    False
2    False
3     True
4    False
Name: B, dtype: bool
    A   B
3  40  35
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
```

You can apply the condition and Boolean masking at the same time using this approach,

```
filtered_df = df[df['A'] > 20]
```

Multiple Conditions

Multiple conditions can be applied using the logical operators(`&` for AND, `|` for OR)

Ex:

1. Selecting rows where column `A` is greater than 20 and column `B` is less than 40.

```python
df = pd.DataFrame({
    'A' : [10, 20, 30, 40, 50],
    'B' : [5, 15, 25, 35, 45]
})

filtered_df = df[(df['A'] > 20) & (df['B'] < 40)]
print(filtered_df)
```

PROBLEMS TERMINAL ··· powershell

```
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals> pytho
    A   B
2  30  25
3  40  35
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
```